ARDUINO PROGRAMMING CHEAT SHEET

SKETCH

Basic Sketch Structure

```
void setup() {
// runs once after each powerup or reset
}

void loop() {
// runs continuously
}
```

Function Definitions

```
<ret. type> <func. name>(<param. type> <param. name>){ … } i.e.:
float circleCircumference(int radius) { return 3.14 * 2 * radius; }
void printGreeting(string name) { Serial.println(name); }
```

VARIABLES, ARRAYS, DATA TYPES

Data Types

```
bool/boolean true false
char -128 - 127
unsigned char 0 - 255
byte 0 - 255
int -32768 - 32767
unsigned int 0 - 65535
word 0 - 65535
long -2147483648 - 2147483647
unsigned long 0 - 4294967295
float -3.4028e+38 - 3.4028e+38
double - same as float except Due
void - indicates no return value
```

Arrays

```
int even[] = {2, 4, 6, 8};
int pins[6];
pins[0] = 10; //indexing from 0
pins[6] = 7; //Common mistake -
indexing from 0 to size - 1 !!!
```

Strings

```
char ex1[3] = {`H','i','\0'};
char ex2[3] = {H','i'};
char ex3[] = „Hi";
char ex4[3] = „Hi";
```

Qualifiers

```
static //persists between func. calls
volatile //in RAM (good for ISR)
const //read-only
PROGMEM //stored in flash
```

Numeric Constants

123	decimal
0b01111110	binary
0123	octal - base 8
0xA2	hexadecimal
123**U**	force unsigned
123**L**	force long
123**UL**	force unsigned long
123.**0**	force float
1.23**e**6	1.23*10^6

OPERATORS

Arithmetic

```
= assignment
+ addition
- subtraction
* multiply / divide
% modulo
```

Comparison

```
== equal to
!= not equal to
<  less than
>  greater than
<= less than or equal
>= greater than or equal
```

Boolean

```
&& and
|| or
! Not
```

Compound Operators

```
++ increment      -- decrement
+= addition       -= subtraction
*= multiplicat.   /= division
```

Bitwise operators

```
& and
| or
^ xor
~ not
<< shift left
>> shift right
```

Compound bitwise operators

```
&= compound bitwise and
|= compound bitwise or
```

Pointer Access

```
& reference: get a pointer
* dereference: get a value
```

BUILT-IN FUNCTIONS

PIN INPUT/OUTPUT

Digital I/O

```
pinMode(pin, mode);
mode - INPUT, OUTPUT, INPUT_PULLUP
int digitalRead(pin);
digitalWrite(pin, state);
state - HIGH, LOW
```

Analog I/O

```
int analogRead(pin);
analogReference(source);
source - DEFAULT, INTERNAL, EXTERNAL
analogWrite(pin, value); //PWM
```

Advanced I/O

```
tone(pin, freq_hz);  noTone(pin);
tone(pin, freq_hz, duration_ms);
byte shiftIn(dataPin, clkPin, order);
shiftOut(dataPin, clkPin, order, val);
bitOrder - MSBFIRST, LSBFIRST
unsigned long pulseIn(pin, state,
timeout); //timeout parameter optional
pulseInLong //same as pulseIn
```

Bits and Bytes

```
byte lowByte(x); byte highByte(x);
byte bitRead(x, bitnumber);
bitWrite(x, bitnumber, bit);
bitSet(x, bitnumber);
bitClear(x, bitnumber);
bit(bitnumber);
```

Math

```
min(x, y); max(x, y);
abs(x); - Absolute value
sin(rad); cos(rad); tan(rad);
sqrt(x); pow(base, exponent);
constrain(x, min, max);
map(val, fromL, fromH, toL, toH);
```

External Interrupts

```
attachInterrupt(interrupt, ISR,
mode);
mode - LOW, CHANGE, RISING,
FALLING
detachInterrupt(interrupt);
interrupts();
noInterrupts();
```

Type Conversions

```
char(val);        byte(val);
int(val);         word(val);
long(val);        float(val);
```

Random Numbers

```
randomSeed(seed);
long random(max); //min = 0
long random(min, max);
```

Time

```
unsigned long millis(); //<50 days
unsigned long micros(); //<70 mins
delay(miliseconds);
delayMicroseconds(useconds);
```

ARDUINO LIBRARIES

Serial - communication via UART

```
begin(long speed);
end();
int available() //num. of bytes
available
int read(); //-1 if none available
int peek(); //read without removing
flush();
print(data); println(data);
write(byte); write(char* str);
write(byte* data, size);
serialEvent();
```

EEPROM.h - non-volatile memory

```
byte read(address);
write(address, byte);
put(addr, data); get(addr, data);
EEPROM[index]; //access as array
```

SoftwareSerial.h - UART on any pin

```
SoftwareSerial(rxPin, txPin);
bool listen(); //only 1 can listen
bool isListening();
begin, read, peek, print, println,
write, available //As in Serial lib.
```

Wire.h - I2C communication

```
begin(); //join a master
begin(addr); //join a slave
requestFrom(address, count);
setClock(clkFreq);
beginTransmission(addr);
write(byte)
write(char* str);
write(byte* data, length);
byte endTransmission();
int available(); //no. of bytes
byte read(); //get next byte
onReceive(handler);
onRequest(handler);
```

Servo.h - control servo motor

```
attach(pin, min_us, max_us);
write(angle); //0 to 180
writeMicroseconds(useconds);
//1000 - 2000; 1500 is midpoint
int read(); //0 to 180 angle
bool attached();
detach();
```

CONTROL STATEMENTS

```
if (x > 0) { … } else { … }
switch (x) {
  case 1:
        …
        break;
  case 2:
        …
        break;
  default:
        …
        break;
}
while (x < 10) { … }
for (int i = 0; i < 10; i++) { … }
do { … } while (x < 10);
break; //Exit loop/switch immediately
continue; //Go to next iteration start
```

Author: Michał Zięba

Adapted from: Mark Liffiton

www.przygodyzkodem.pl
GITHUB: przygodyzkodem
IG: przygodyzkodem
FB: przygodyzkodem